SUPPORTING THE EVOLUTION OF DESIGN ARTIFACTS WITH REPRESENTATIONS OF CONTEXT AND INTENT

Gerhard Fischer[1], *Kumiyo Nakakoji*[1,2] *and Jonathan Ostwald*[1]

[1]Center for LifeLong Learning and Design (L3D)
Department of Computer Science and Institute of Cognitive Science
University of Colorado
Boulder Colorado 80309-0430, USA

[2]Software Engineering Laboratory
Software Research Associates, Inc.
1-1-1 Hirakawa-cho, Chiyoda-ku, Tokyo 102, Japan

tel: (303) 492-7514 fax: (303) 492-2844
E-mail: {gerhard, kumiyo, ostwald}@cs.colorado.edu

ABSTRACT
The design of complex artifacts is essentially an evolutionary process that requires collaboration among stakeholders. Domain-oriented design environments (DODEs) support the evolution of artifacts both by individual designers and by designers participating in long-term, indirect collaboration. DODEs provide representations for generic and specific levels of context. This context supports individual designers by making the information space relevant to the current design intent, and long-term collaboration among designers by allowing them to ground their communication around design artifacts. We demonstrate our approach using the KID (Knowing-in-Design) system, articulate principles for representations of context and intent, and discuss various approaches to represent intent and context in design environments.

KEYWORDS: domain-oriented design environments, shared context, explicit representations for intent, communication of intent, evolution of design artifacts, knowledge-based information delivery, long-term indirect collaboration

1. Introduction

The design of complex artifacts is essentially a collaborative and ongoing process. The need for collaboration stems from the fact that the knowledge required to understand and solve problems spans many fields and is distributed among many stakeholders[1] [10, 26]. The need for ongoing design stems from the fact that complex design problems are ill-structured [24], meaning that one cannot completely understand design requirements before making significant steps toward the solution. There will always remain some requirements that can only be recognized when the artifact is used [13]. Design theorists advocate iterative problem-solving processes that emphasize communication between designers and are grounded by design representations. For example, Rittel [20] views design as an argumentative process, and Schoen [21] sees design as a conversation with the materials of the design.

Complexity in design arises from the need to synthesize different perspectives on a problem, to manage large amounts of information potentially relevant to a design task, and to understand the design decisions that have determined the possibly long-term evolution of a designed artifact. Our approach to supporting design with computers focuses on human-centered design support with domain-oriented design environments (DODEs). Design environments are human-computer collaborative problem-solving systems [25] that provide (1) design media and tools with which designers can represent their design, and (2) intelligent agents that support designers in using the system's design knowledge for understanding and reflecting upon their emerging design artifacts. Rather than modeling the cognitive processes of designers, DODEs augment the abilities of designers to understand, manage, and communicate complexity.

This paper argues that design environments must enable the evolution of design artifacts by supporting collaboration

Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee.
DIS 95 Ann Arbor MI USA © 1995 ACM 0-89791-673-5/95/08..$3.50

[1]Stakeholders of a design task include people with various roles, such as designers, clients, and end-users. To focus our discussion on issues of collaboration between people and computers, rather than among different roles, we use the term "designers" to refer to stakeholders in general, and do not distinguish among the different roles of stakeholders.

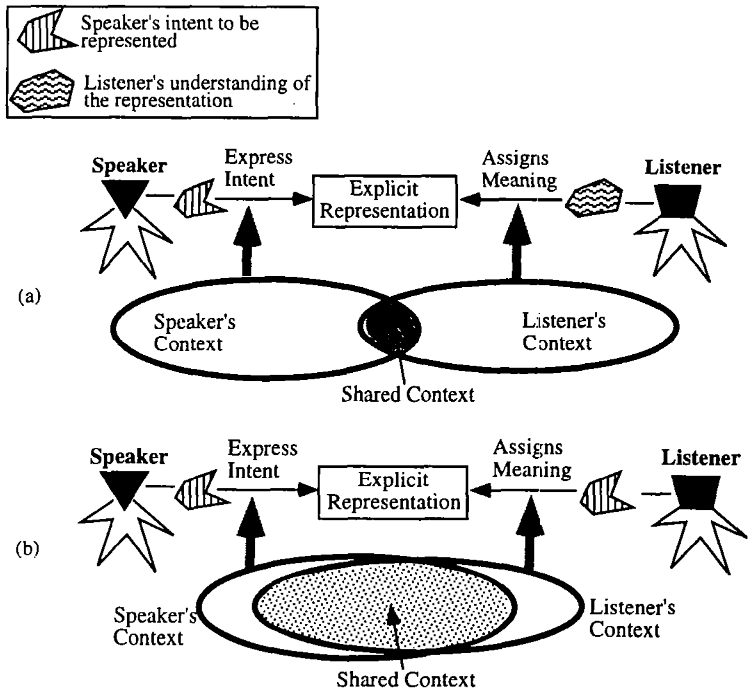


Figure 1: The Shared Context and Communication of Intent in Design

People use background context to represent intent and understand the representation. If a speaker and listener have different context, communication breakdown occurs because a listener assigns a meaning to a representation that is different from the original meaning of the representation the speaker intended. The more context they share, the smaller the communication gap becomes.

between designers. Design environments support collaboration by serving as a media for communication. Because design environments house the evolving artifacts, they provide a context for communication. As representations to communicate intent accumulate in the design environment, they, in turn, contribute to the evolution of context that supports subsequent collaboration.

In this paper, we first describe the evolution of design artifacts and the necessity of communication of design intent. Then, we discuss how our DODEs support the communication of intent with an example scenario using the KID (Knowing-in-Design) design environment. Section 4 discusses issues in representations for context and intent and presents other approaches.

2. Evolution of Design Artifacts Through Communication of Intent

This section takes a close look at the relation between the evolution of design artifacts and the communication between collaborative designers. In particular, we look at communication in terms of *intent* (the meaning behind the message) and *context* (the background against which the message is articulated and understood). In what follows, we first describe everyday communication, then describe communication in design, mediated by design environments.

In everyday interpersonal communication, intent is often casually articulated and understood against a rich background of shared experience and circumstances. To describe the process of communication of intent, we use the terms "speakers" and "listeners" to refer to two roles: those who articulate their intent and those who try to understand (assign a meaning to) the articulated representation, respectively.

Speakers express their intent by implicitly using background context. Listeners, using their own context, try to assign a meaning to the representation. Ideally, the speaker's intent behind the representation and the assigned meaning by the listener should be the same. However, if the listener uses a context that is different from the the one that the speaker used (and this is true for most cases) when interpreting the representation, mis-communication occurs.

As depicted in Figure 1, each stakeholder has a different context, as each person has a different understanding of a problem. Such context is vague and cannot be completely described or expressed. Communication breakdown occurs when a speaker and a listener have little shared context because the assigned meaning to the representation by a listener mismatches to the original meaning of the representation that the speaker intended. As illustrated in Figure 1-(b), the more context the speaker and the listener share (i.e., intersection), the less mismatch between the speaker's intent and assigned meaning by the listener.

In contrast to everyday interpersonal communication, designers express their intentions using explicit design representations. According to Schoen's theory, designers work in an alternating cycle of action and reflection [21]. The designer *acts* to shape the design situation by creating or modifying design representations, and the situation "talks back" to the designer, revealing unanticipated consequences of the design actions. In order to understand the situation's back-talk, the designer *reflects* on the actions and consequences, and plans the next course of action. Thus, designers are speakers when they act on a design representation, and listeners when they reflect on the representation. This interaction between designers as speakers and designers as listeners drives the evolution of artifacts.

Complex design may span over a long period of time, and even expert designers cannot attend to all facets of a complex design task. Externally articulating a partially understood design intention is equally as important as externalizing the partial solution. As Sharples has noted, making ideas explicit "is not a matter of emptying out the mind but of actively reconstructing it, forming new associations, and expressing concepts in linguistic, pictorial, or any explicit representational forms" [22]. When designers pause for reflection, they are directing attention toward an intimate relationship between the partially understood problem and solution. Because intentions become context for subsequent design moves, their original meaning are easily lost if not made explicit.

Computer support for ongoing and collaborative design must provide representations of both intent and context in order for meaning to be communicated from speaker to listener over time. Representations created to communicate intent in the past can be reused as context to understand design problems in the present. The representations that are created to communicate intent in the present become part of the evolving context for the future because they are attached to artifacts. As part of the evolving context, these representations support interpretation of subsequent communication, which, in turn, also becomes part of the evolving context. Thus, the evolution of design artifacts is both driven by, and supported by, communication of intent.

In summary, supporting the communication of intent in design environments requires that (1) intent is explicitly represented, and (2) that representations of intent are accumulated and reused as context for understanding subsequent communication of intent. The next section describes how our DODEs support this communication of intent.

3. DODEs: Explicit Representation of Context

In our work, we have created collaborative systems named *domain-oriented design environments* in support of design. As the name suggests, DODEs are built to support design in a specific domain, such as kitchen design or LAN design.

DODEs are built upon a domain-independent architecture that provides a structure for domain knowledge, and mechanisms for delivering knowledge as it is needed to support design. We have developed our domain-independent architecture through numerous attempts to create domain-oriented design environments (for details see [3]). The architecture consists of the following five components: (1) a construction component, (2) an argumentation component, (3) a catalog of interesting design examples, (4) a specification component, and (5) a simulation component. The individual components are linked by knowledge-based mechanisms: a construction analyzer (built as a critiquing system [6]), an argumentation illustrator and a catalog explorer.

DODEs instantiate the architecture for a particular design domain. A DODE instantiated to support kitchen design, for example, contains a construction component for constructing a floor layout, a specification component for specifying abstract kitchen characteristics, an argumentation component containing issue-based information relevant to kitchen design, and a catalog for storing kitchens designed using the system. The construction and specification components are used by the designer to express intentions, or to make design moves, for a given design task. As the design task proceeds, the states of the construction and specification form an explicit context that is shared between designer and DODE.

DODEs provide feedback to designers as they design, rather than requiring designers to construct a final product before receiving feedback. In this way, DODEs help designers evolve designs and understand the effects of individual design moves. The interaction between a designer and a DODE can be seen as a conversation in which the designer speaks by making a design move and listens to the feedback provided by the environment. Conversely, the DODE listens to the designer's design moves and speaks by providing feedback. The history of the design process, including the designer's moves and the DODE's feedback, forms a shared and evolving context that grounds communication between designer and DODE.

In what follows, we first describe how DODEs support generic and specific levels of context, and then we describe how DODEs use the context to support both individual and collaborative aspects of design through communication of intent. A demonstration of our approach using the KID system follows.

3.1. Two Levels of Context

DODEs provide an explicit context for design at two levels: (1) at a generic level, the DODE is a domain-level substrate for designing artifacts, and (2) a specific level, the context becomes specialized according to the individual artifact being designed.

Because a DODE is tuned to support design of artifacts for a particular domain, there is a generic context that provides the initial basis for communication between designers and a DODE. This generic context consists of all the domain knowledge in the DODE at a given time. Because each design task is in some sense unique, it is not possible to perfectly anticipate the intentions of designers who use the DODE. Each artifact designed using the DODE has the potential to add to the cumulative generic context. Thus, the generic level of context evolves as the DODE is used to design artifacts.

When designers first begin a design, the DODE can interpret the designers' actions only against the generic context. As designers construct the design, they are incrementally representing their intentions for *this* artifact. As the design progresses, the designers make more and more intent explicit, and the shared context for the specific design task becomes more specific. Thus, the specific level of context evolves as designers make design moves toward the design of an artifact.

3.2. The Use of Shared Context

The two levels of context provided by DODEs enables support for designer-computer communication, and for long-term communication between designers:

- specific level context supports designer-computer communication by making the system's feedback relevant to the current design intent, and
- generic level context supports long-term collaboration among designers by grounding their communication around design artifacts [19].

DODEs offer great capacity for storing large volumes of information and integrating diverse information sources, such as solutions to previous design problems and collections of argumentation. However, access to large information spaces creates a new problem for designers: information overload. In situations of information overload, the critical resource for designers is not information, but rather the attention needed to process information. Therefore, when presenting designers with information, the primary concern is to present items that are relevant to the task at hand [9].

The specific context defined by the construction and specification components allow the system to provide information relevant to a dynamic context that is shared by the designer and the design environment. This shared context enables precise intervention by the system's knowledge delivery mechanisms (e.g., critics [6]), reduces annoying interruptions, and increases the relevance of information delivered to designers.

In addition to supporting individual designers to design, DODEs support long-term indirect collaboration between designers [5]. Long-term collaboration is required in the design of complex and evolving artifacts, which are maintained and modified over a span of years. Such artifacts are not designed from scratch but are iteratively refined. In this sense, design artifacts are never complete but instead are constantly evolving.

As designers use a DODE for many design tasks over a long period of time, design artifacts as well as representations for intent are accumulated in repositories of the DODE. These repositories allow designers to collaborate indirectly with past designers by utilizing the generic domain-level context in the form of information and artifacts from prior designs.

In summary, in our DODEs, communication of intent takes place around design artifacts. By capturing the intentions and priorities of past designers and associating them with artifacts, design environments are able to locate stored artifacts and information that are relevant to the current designer's intention, providing a rich context for assessing the relevance of delivered information.

3.3. An Example: the Use of KID

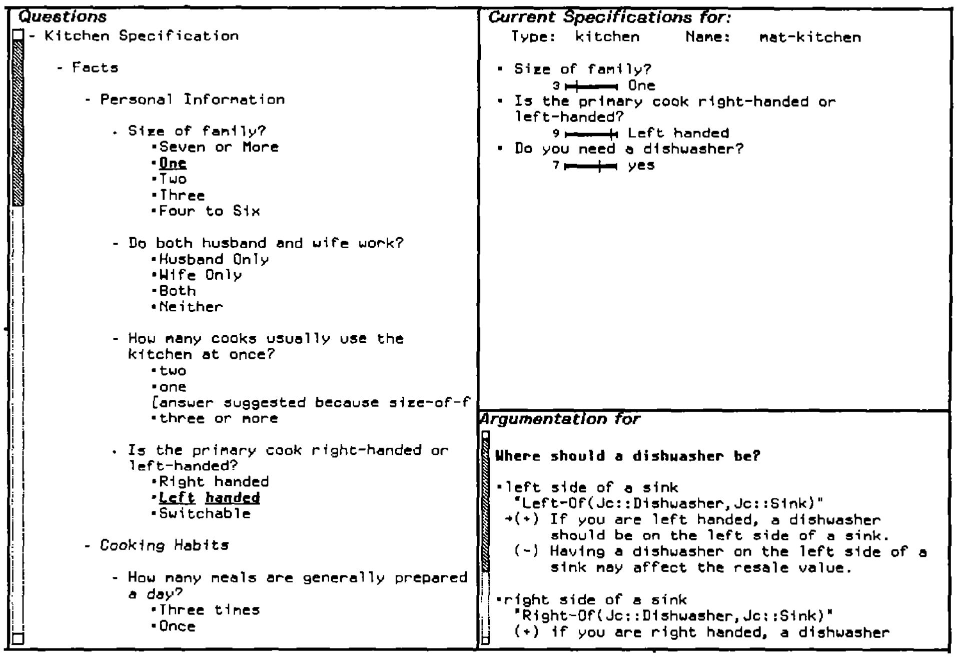
The KID system [15] is a design environment for creating kitchen floor plans that substantially extends the JANUS system [8]. Figures 2 and 3 show screen images of the KIDSPECIFICATION and KIDCONSTRUCTION components of KID. The specification component supports designers in framing their design problem; i.e., specifying design goals, objectives, and criteria or constraints. The construction component supports designers in constructing the solution form (a floor plan) of the design artifact.

The following scenario illustrates how KID uses context to support designer-computer interaction as well as long-term indirect collaboration between designers.

A kitchen designer, Jane, specifies requirements for her design task using KIDSPECIFICATION (Figure 2), and starts constructing a floor plan using KIDCONSTRUCTION (Figure 3). When she puts a dishwasher on the right side of a double-bowl sink, critic messages appear on the screen, one of which notifies her that she should put the dishwasher on the *left* side of the sink (see the *Message* window in Figure 3). Wondering why, she clicks on the critic message.

The corresponding argument is presented, stating that this kitchen should have the dishwasher on the left side of the sink because she specified that this kitchen is for a left-handed cook (see Figure 2). Jane understands this suggestion, but at the same time, she notices the additional argument stating that a dishwasher on the left side of a sink may affect the resale value (see the *Argumentation* window in Figure 2). Realizing that the resale value of the kitchen is actually a very important concern, she adds this requirement using KIDSPECIFICATION, and leaves the dishwasher on the right side of the sink. When she completes her design, Jane adds it to KID's catalog of kitchen designs.

Later, another designer, Bob, uses KID to design a kitchen. Using KIDSPECIFICATION, Bob specifies that he would like to design a kitchen for a left-handed cook and starts his design. When he places the dishwasher on the right side of the sink, the same critic rule fires, notifying him that the dishwasher should be to the left side of the sink. He looks at the catalog window, where KID has presented Jane's kitchen as an artifact relevant to his design. He notices that Jane's kitchen has the dishwasher to the right of the sink (not shown). Wondering why KID presented Jane's kitchen as a relevant example, which also violates the critic rule, Bob looks at the specification for her design, and sees that for Jane's design task, the resale value was more important than left-handedness.



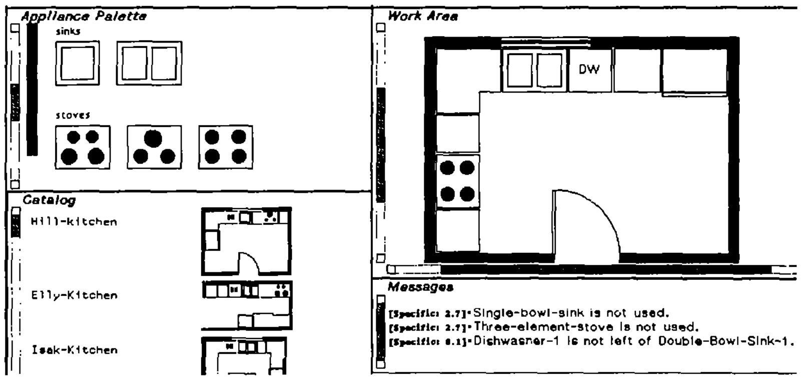
```
Questions
- Kitchen Specification
    - Facts
        - Personal Information
            . Size of family?
                •Seven or More
                •One
                •Two
                •Three
                •Four to Six
            - Do both husband and wife work?
                •Husband Only
                •Wife Only
                •Both
                •Neither
            - How many cooks usually use the
              kitchen at once?
                •two
                •one
                [answer suggested because size-of-f
                •three or more
            . Is the primary cook right-handed or
              left-handed?
                •Right handed
                •Left handed
                •Switchable
        - Cooking Habits
            - How many meals are generally prepared
              a day?
                •Three times
                •Once
```

```
Current Specifications for:
  Type: kitchen     Name:  mat-kitchen

 • Size of family?
        3 One
 • Is the primary cook right-handed or
   left-handed?
        9 Left handed
 • Do you need a dishwasher?
        7 yes
```

```
Argumentation for

Where should a dishwasher be?

•left side of a sink
 "Left-Of(Jc::Dishwasher,Jc::Sink)"
 →(+) If you are left handed, a dishwasher
      should be on the left side of a sink.
  (-) Having a dishwasher on the left side of a
      sink may affect the resale value.

•right side of a sink
 "Right-Of(Jc::Dishwasher,Jc::Sink)"
  (+) if you are right handed, a dishwasher
```

Figure 2: KIDSPECIFICATION

The user interface is based on the questionnaire forms used by professional kitchen designers to elicit their clients' requirements. KIDSPECIFICATION provides an extensible collection of questions (issues) and alternative answers from which designers select the requirements associated with their current design intent (see the *Questions* window). The summary of currently selected answers appears in the *Current Specifications for* window, and designers can assign weights to the selected answers to represent the relative importance of the specified requirements. If no existing alternatives express their position, designers can add or modify information in the underlying argumentation base. *Argumentation for* window provides further explanation about how a presented critic message (i.e., a location of a dishwasher with regard to a sink) (see Figure 3) is related to the current specification (i.e., one of the selected answers - a left-handed cook), as well as alternatives for the location of a dishwasher.



```
[Specific: 2.7]•Single-bowl-sink is not used.
[Specific: 2.7]•Three-element-stove is not used.
[Specific: 8.1]•Dishwasher-1 is not left of Double-Bowl-Sink-1.
```

Figure 3: KIDCONSTRUCTION

Designers construct a kitchen floor plan in the *Work Area* using a direct manipulation style to select and place design units from the appliance palette. Designers may copy an example from the *Catalog* window, where catalog examples are presented in the order of accordance with the current specification (see Figure 2). The *Messages* window presents critiquing messages that are detected by KID. Numbers indicate computed relative importance of each critiquing message in terms of the current specification.

3.4. Discussion

Support for Designer-DODE communication. In KID, the designer's intention is articulated by manipulating interface objects in the construction and specification components. The specification provides information about the designer's high-level intentions. From the construction, the system obtains information about the design moves that have been made, which represent the designer's solution-level intentions.

The system uses the cumulative state of the specification and construction components as the current specific context. The representations in the specification and construction that define the specific context are *shared* because the state of the representations is accessible to both the designer and the system.

In the scenario, the specific context for Jane's design included that she wanted to design a kitchen for a left-handed person, and that she placed the dishwasher on the right side of the sink. Because KID provides explicit representations for both specification and construction, it is able to provide feedback specific to Jane's design situation. If KID provided support for only construction, it couldn't have provided feedback relevant to Jane's high-level intentions.

KID uses computational critic mechanisms [6] to alert designers to problematic design situations, such as a violation of domain design rules, and to provide information relevant to the situation. This mechanism allows designers to become aware of implications of the current design context in which they are engaged. Using the shared specific context, KID was able to detect a conflict between Jane's high-level intentions (left-handed) and solution-level intentions (dishwasher on right side of sink), and to point Jane toward the realization that designing a kitchen for a left-handed person would sacrifice resale value.

Support for Long-term Collaboration. KID contains two collections of domain knowledge: an argumentation base that stores design rationale and a catalog base that stores design artifacts. The argumentation base is a semi-structured design space that expresses interdependencies between design decisions as well as the contexts in which the interdependencies are relevant. The catalog base contains precedent design cases represented as a construction (floor plan) and a specification (design requirements), which are created by designers in the past. Thus, the domain knowledge is a generic domain-level context that allows users of KID to collaborate over a long period of time.

In the scenario, Bob and Jane communicated indirectly in the sense that Jane's design was placed into the catalog base and subsequently delivered by KID to Bob as a design relevant to his task. Jane's kitchen added to the domain knowledge stored in KID, thereby increasing the generic domain-level context. Although not illustrated in the scenario, Jane might also have added new arguments to the argumentation base, rather than agreeing with an argument that was already in the argumentation.

Mechanisms. KID uses *specification-linking rules* to map from a preference articulated in the specification to a corresponding combination of constraints that should be satisfied in the construction. The specification-linking rules enable KID to detect design situations in which the construction and specification are in conflict. Such conflicts are brought to the designer's attention by two knowledge-delivery mechanisms:

- RULE-DELIVERER locates information in the argumentation base corresponding to the conflict between the specification and construction detected through *specific critics* [6]. The argumentative information helps designers to understand the problem and alternative means for resolving it. In the scenario, RULE-DELIVERER detected a conflict between Jane's desire to design a kitchen for a left-handed person and her placement of the dishwasher to the right of the sink.
- CASE-DELIVERER orders the catalog space so that examples relevant to the current design situation are easily accessible to the designer [16]. CASE-DELIVERER computes the conformity of each catalog example to the current partial specification by (1) applying specific critics to each catalog example, (2) computing an appropriateness value for the example as the weighted sum of the critic evaluations, (3) ordering the examples according to the values, and (4) presenting the ordered catalog examples. In the scenario, CASE-DELIVERER presented Jane's design to Bob because both designers specified that they wished to design a kitchen for a left-hander.

In summary, KID's explicit representations of a problem specification and solution construction help designers to achieve and maintain a common understanding of the problem and prevent them from overlooking important considerations. KID uses these representations as the current context to identify task-relevant information. KID's knowledge bases contain design information and artifacts accumulated through past design efforts, enabling designers to collaborate indirectly with their peers from the past.

The approach described here in terms of KID has demonstrated how the representations of specification and construction of a DODE serve as a context to facilitate communication of intent between designers and a system, and among designers, for a relatively mature, stable domain such as kitchen design. For *immature* or *unstable* domains, which are relatively new, still under exploration, or heavily dependent on state-of-the-art technologies, it is difficult to to identify and design such representations. In the next section, we discuss other approaches to represent design intent and context in DODEs.

4. Issues in Representations for Context and Intent

The representation of design context and intent is not a well-defined problem, and constructing such representations (i.e., a DODE itself) is yet another design task. We have constructed DODEs for varieties of design domains including user-interface design, kitchen design, LAN design, voice-dialogue design, and software design. With each prototype, we have studied a variety of representational formalities.[2] On the one hand, the more formally designers represent their intent, the better understanding of the intent the DODE will have and, consequently, the more context the system and the designers share. On the other hand, imposing the machine's formality on designers forces them to represent design actions in an unfamiliar language and thus undermines their expressive ability.

We have identified the following requirements for representations of context and intent that both DODEs and designers can use:

[2]By *formality*, we mean the degree to which the system can interpret the semantics (content) of the representation.

- *expressive:* Designers must be able to represent intended concepts directly and distinctly using familiar notations and languages;
- *associative:* Designers and/or DODEs must be able to associate representations with those for related concepts. "What something *means* lies in how it connects to other things we know" [14]. By providing links, relations, and connections among multiple representations, designers gain an understanding of the content of the design and the partial design task.

Construction kits and critiquing rules are considered to be formal representations because the association of the representation is automatically done by the system. Argumentation is a semi-formal representation where informal textual and graphical representations are linked by designers.

We have developed the Seeding — Evolutionary Growth — Reseeding (SER) model to support the gradual development and refinement of representational formalisms [7]. In the model, the evolution of a DODE (i.e., representations of a DODE) is driven by its use in designing individual artifacts, which create new requirements. Explicit representations of context serve not only evolution of individual design artifacts but also the evolution of DODEs themselves.

In what follows, we describe several forms for a DODE for representing design intent. The first four representations are associative by systems, and the last one is associative by designers supported by systems.

The Construction. The representation in a construction component is a formal design form. A DODE provides a palette of objects pre-assigned with domain semantics and a workplace where a user can manipulate those objects in order to construct a design solution. This representation can be "parsed" by the system, providing the system with information about the artifact under construction. For example, with KID (as presented in Section 3.3), the system knows that a rectangle displayed in the workplace with a label "DW" represents a dishwasher, and that the dishwasher is next to a double-bowl sink, an adjacent rectangle with two smaller rectangles inside.

The representation for the design solution is interpretable into a single meaning and can be represented formally within a computer system. Research in formal approaches in the AI in Design field [2] studies the interpretation of such solution representation.

The Specification. A specification component provides informal representations associated with formal rules. In some design domains, a set of natural language statements exists to represent goals, objectives, and constraints of the task, shared by a community of designers. In the kitchen design, for example, a questionnaire is used to elicit a client's requirements. In the LAN design, there are a set of typical questions asked by expert network designers before installing a network.

Such statements are associated with design decisions, and the same associations are used repeatedly in many design tasks within the domain. The interdependencies among those design decisions are captured as design rationale, or "arguments." As discussed in the previous section, KID provides an argumentation base that is based on the IBIS structure [1]. Some of design decisions (i.e., answers in the IBIS) and interdependencies (i.e., arguments in the IBIS) are associated with predefined predicates over the construction, as described above. KIDSPECIFICATION allows designers to select some answers in the argumentation base, and KID infers interdependencies using the partial specification. These interdependencies are used as *specification-linking rules* to identify relevant critiquing rules (i.e., "specific critics"; see [6]) and relevant reusable design examples [16].

Sketches. Sketches are graphics drawn by designers that can be parsed by a DODE. Designers use drafting paper and pencil to gain their own understanding about the design problem. Most of existing design drawing systems do not allow designers to deal with abstract, vague, or uncertain properties of a partial design. Designers do not feel comfortable in using such tools, especially at the initial stages of design. And yet, such rough drafting conveys very important information regarding design intent. For example, we observed that a kitchen designer drew several bubbles with labels such as *working center, cooking center,* and *storage area* on a piece of paper at the very beginning of her design process. The sketch gave her a rough idea of how the workflow might be in the design without worrying about detailed precise dimensions of each appliance.

It would have been possible to ask the designer to formally represent her design intention, such as the cooking center should be adjacent to the storage area. However, in general, designers will not expend the effort until they see the benefit of entering it into the computer [4]. Supporting early design in the way that designers are accustomed to doing it (i.e., such as drafting) enables DODEs to bear at a time when they can have the least cost and the most impact [11].

Gross et al. have attempted to use hand-drawn diagrams to index architectural design cases as well as retrieve useful design cases. The underlying Electronic Cocktail-Napkin system recognizes the elements of a diagram and interprets them in the context of the architectural design domain [12]. This syntactic analysis of a graphic has enabled the DODE for symbolic and numerical analysis such as critiquing, simulation, visualization, and retrieval of relevant cases. The eMMa system, a design environment for multimedia authoring [17], also provides a mechanism to retrieve images by analyzing pictorial data from a library of 1,000 images. The system uses free-hand writing by a user as a query to match the graphic by tracing the border of images (see Figure 4). The system also allows users to specify a theme color and retrieve images that have similar hues.

History. The INDY network design system [19] supports representations of the history of design artifact. History provides the background context of design decisions in terms of the temporal relationship. The order of design

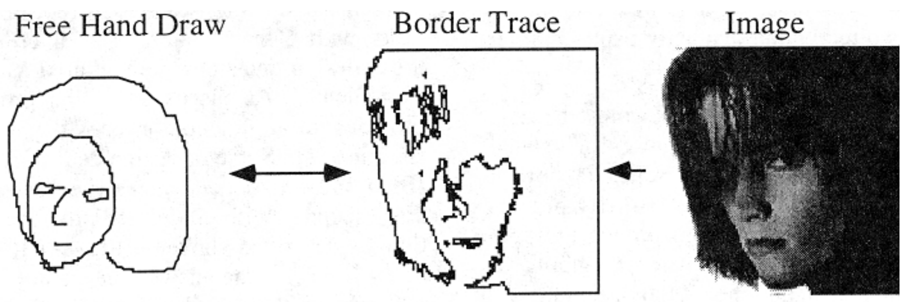


Figure 4: Retrieval of Images using Free-hand Drawing in eMMa

decisions made implies prioritization of dependencies and thus plays a crucial role in understanding the design. The INDY system keeps track of all the design decisions made by whom over a network design, and allows designers to play backward and forward to simulate how the design has evolved.

Descriptive Annotations. The INDY system discussed above allows designers to annotate a network design. Annotations are written in natural language and provide design rationale for associated design decisions.

The XNETWORK system [23] for LAN design supports designers to associate electronic mail to a network design. In the LAN network design, change requests and bug reports are communicated through e-mail, which drives long-term evolution of the design. The system supports designers in incrementally formalizing the accumulated information. The system suggest to designers to which design objects an e-mail is likely to be related by parsing the e-mail and inferring attribute values for predefined attributes of design objects, such as machine types, names, users, capacity, and performance. Using this information, network designers make a final decision on how to structure the information space.

The EVA system [18] is a hypermedia substrate that allows system analysts, software designers, and end-users to collaboratively evolve software prototypes. The system integrates executable prototypes and descriptive representations, such as text, graphics, and e-mail. Users of EVA can assign semantics to an association among representations. Multiple designers gradually evolve the design space by visiting representations constructed by other designers, adding their design intent and understanding of their own, and associating them with the existing information. Using EVA, mediating collaboration, design intent, solution, and context can coevolve by multiple designers.

5. Conclusion

This paper presented our domain-oriented design environment approach and demonstrated how the evolution of design artifacts can be supported with explicit representations of design intent. We have identified that such representations need to be expressive and associative in order to be useful both to designers and to the knowledge-based design support mechanisms in design environments. Because designers explicitly articulate their design intent and build design artifacts in DODEs, these representations can be reused as context for reflecting upon a partially constructed design as well as understanding previous design cases. A challenge is to identify appropriate representations for a given design domain. The formalisms described in the previous section are by no means an exhaustive enumeration of possible representations for design intent. Rather they should be viewed as a starting point for future research into representations of context and intent in design.

Acknowledgments

We thank the people at the Center for LifeLong Learning and Design (L3D) at the University of Colorado, who contributed to the conceptual framework and the systems discussed in this paper. The research was supported by the National Science Foundation under grants No. IRI-9015441 and MDR-9253425; Software Research Associates, Inc. (Tokyo); NYNEX Science and Technology Center; the Colorado Advanced Software Institute; US WEST Advanced Technologies; and the National Science Foundation and the Advanced Research Projects Agency under Cooperative Agreement No. CDA-940860.

REFERENCES

1. J. Conklin, M. Begeman. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. *Transactions of Office Information Systems 6*, 4 (October 1988), 303-331.

2. R.D. Coyne, M.A. Rosenman, A.D. Radford, M. Balachandran, J. Gero. *Knowledge-based Design Systems.* Addison Wesley Publishing Company, Reading, MA., 1989.

3. G. Fischer. Domain-Oriented Design Environments. *Automated Software Engineering 1* (1994), 177-203.

4. G. Fischer, A.C. Lemke, R. McCall, A. Morch. Making Argumentation Serve Design. *Human Computer Interaction 6*, 3-4 (1991), 393-419.

5. G. Fischer, J. Grudin, A.C. Lemke, R. McCall, J. Ostwald, B.N. Reeves, F. Shipman. Supporting Indirect, Collaborative Design with Integrated Knowledge-Based Design Environments. *Human Computer Interaction, Special Issue on Computer Supported Cooperative Work 7*, 3 (1992), 281-314.

6. G. Fischer, K. Nakakoji, J. Ostwald, G. Stahl, T. Sumner. Embedding Critics in Design Environments. *The Knowledge Engineering Review Journal 8*, 4 (December 1993), 285-307.

7. G. Fischer, R. McCall, J. Ostwald, B. Reeves, F. Shipman. Seeding, Evolutionary Growth and Reseeding: Supporting Incremental Development of Design Environments. *Human Factors in Computing Systems, CHI'94 Conference Proceedings (Boston, MA)*, ACM, 1994, pp. 292-298.

8. G. Fischer, R. McCall, A. Morch. JANUS: Integrating Hypertext with a Knowledge-Based Design Environment. *Proceedings of Hypertext'89 (Pittsburgh, PA)*, ACM, New York, November, 1989, pp. 105-117.

9. G. Fischer, K. Nakakoji. Beyond the Macho Approach of Artificial Intelligence: Empower Human Designers - Do Not Replace Them. *Knowledge-Based Systems Journal 5*, 1 (1992), 15-30.

10. J. Greenbaum, M. Kyng (Eds.). *Design at Work: Cooperative Design of Computer Systems.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.

11. M. Gross. Computers as Cocktail Napkin: Recognizing and Interpreting Hand Drawn Diagrams in Design. *Proceedings of the Conference on Advanced Visual Interface*, Bari, 1994.

12. M. Gross, C. Zimring, E.Do. Using Diagrams to Access a Case Base for Architectural Designs. J. S. Gero, F. Sudweeks (Ed.), *Artificial Intelligence in Design'94*, Kluwer Academic Publishers, 1994, pp. 129-144.

13. A. Henderson, M. Kyng. There's No Place Like Home: Continuing Design in Use. In J. Greenbaum, M. Kyng (Eds.), *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1991, Chap. 11, pp. 219-240.

14. M. Minsky. Logical Versus Analogical or Symbolic Versus Connectionist or Neat versus Scruffy. *AI Magazine 12*, 2 (Summer 1991), 35-51.

15. K. Nakakoji. *Increasing Shared Understanding of a Design Task Between Designers and Design Environments: The Role of a Specification Component.* Ph.D. Thesis, Department of Computer Science, University of Colorado, Boulder, CO, 1993. Also available as TechReport CU-CS-651-93.

16. K. Nakakoji. Case-Deliverer: Retrieving Cases Relevant to the Task at Hand. In S. Wess, K. Althoff, M.M. Richter (Eds.), *Lecture Notes in Artificial Intelligence: Topics in Case-Based Reasoning: Selected Papers from EWCBR-93*, Springer-Verlag, Kaiserslautern, Germany, 1994, pp. 446-470.

17. K. Nakakoji, B.N. Reeves, A. Aoki, H. Suzuki, K. Mizushima. *eMMaC: Knowledge-Based Color Critiquing Support for Novice Multimedia Authors.* Submitted to Multimedia'95.

18. J. Ostwald. *The Evolving Artifact Approach for Building Knowledge Systems in Information Intensive Domains.* Department of Computer Science, University of Colorado, Boulder, CO, August, 1993.

19. B.N. Reeves, F. Shipman. Supporting Communication between Designers with Artifact-Centered Evolving Information Spaces. *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'92)*, ACM, New York, November, 1992, pp. 394-401.

20. H.W.J. Rittel. Second-Generation Design Methods. In N. Cross (Ed.), *Developments in Design Methodology*, John Wiley & Sons, New York, 1984, pp. 317-327.

21. D.A. Schoen. *The Reflective Practitioner: How Professionals Think in Action.* Basic Books, New York, 1983.

22. M. Sharples. Cognitive Support and the Rhythm of Design. In T. Dartnall (Ed.), *Artificial Intelligence and Creativity*, Kluwer Academic Publishers, Netherlands, 1994, pp. 385-402.

23. F. Shipman, R. McCall. Supporting Knowledge-Base Evolution with Incremental Formalization. *Human Factors in Computing Systems, CHI'94 Conference Proceedings*, ACM, 1994.

24. H.A. Simon. *The Sciences of the Artificial.* The MIT Press, Cambridge, MA, 1981.

25. L.G. Terveen. An Overview of Human-Computer Collaboration. *Knowledge-Based Systems Journal* (1995). (in press).

26. D. Walz, J. Elam and B. Curtis. Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration. *CACM 36*, 10 (October 1993), 63-77.